EAP-based Authentication Service for CoAP
draft-marin-ace-wg-coap-eap-06

Abstract

   This document describes an authentication service that uses EAP
   transported by means of CoAP messages with two purposes:

   o Authenticate two CoAP endpoints.

   o Bootstrap key material to protect CoAP messages exchanged between
      them.

Status of This Memo

   This Internet-Draft is submitted in full conformance with the
   provisions of BCP 78 and BCP 79.

   Internet-Drafts are working documents of the Internet Engineering
   Task Force (IETF). Note that other groups may also distribute
   working documents as Internet-Drafts. The list of current Internet-
   Drafts is at https://datatracker.ietf.org/drafts/current/.

   Internet-Drafts are draft documents valid for a maximum of six months
   and may be updated, replaced, or obsoleted by other documents at any
   time. It is inappropriate to use Internet-Drafts as reference
   material or to cite them other than as "work in progress."

   This Internet-Draft will expire on April 27, 2018.

Table of Contents

1. Introduction

   The goal of this document is to describe an authentication service
   that uses the Extensible Authentication Protocol (EAP) [RFC3748].
   The authentication service is built on top of the Constrained
   Application Protocol (CoAP) [RFC7252] and allows authenticating two
   CoAP endpoints by using EAP without the need of additional protocols
   to bootstrap a security association between them.

   In particular, the document describes how CoAP can be used as EAP
   lower-layer [RFC3748] to transport EAP between a CoAP server (EAP
   peer) and the CoAP client (EAP authenticator) using CoAP messages.
   The CoAP client may contact with a backend AAA infrastructure to
   complete the EAP negotiation as described in the EAP specification
   [RFC3748].

The assumption is that the EAP method transported in CoAP MUST generate cryptographic material [RFC5247] . In this way, the CoAP messages can be protected. There are two approaches that we have considered in this document:

o To define a new AUTH option that includes an authentication tag generated with the cryptographic material exported during an EAP authentication. This new option is used to protect the integrity of the CoAP message that carries the AUTH option. (NOTE: Encryption will be considered in the future).

o To establish a DTLS security association using the exported cryptographic material after a successful EAP authentication. [I-D.ohba-core-eap-based-bootstrapping]

This document also provides some comments about implementation of a proof-of-concept of this preliminary idea

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. General Architecture

The Figure 1 shows the architecture defined in this document. Basically a node acting as the EAP peer wants to be authenticated by using EAP. At the time of writing this document, we have considered a model where the EAP peer will act as CoAP server for this service and the EAP authenticator will act as CoAP client and may interact with a backend AAA infrastructure. Nevertheless, a model where the EAP peer act as CoAP client and the EAP authenticator as CoAP server will be also analyzed in the future.

```
         +------------+         +------------+         +--------------+
         | EAP peer/  |         | EAP auth./ |         | EAP server/  |
         | CoAP server|+------+| CoAP client|+-----+|  AAA server  |
         +------------+  CoAP  +------------+  AAA  +--------------+
```
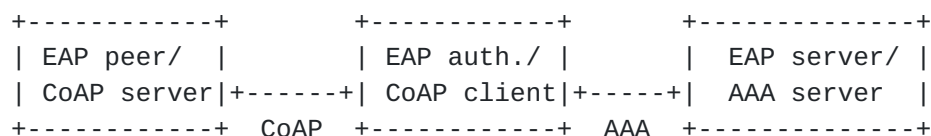
Figure 1: CoAP EAP Architecture

3. General Flow Operation

The authentication service uses the CoAP protocol as transport layer for EAP. CoAP becomes an EAP lower-layer (in EAP terminology). In general, it is assumed that, since the EAP authenticator may need to

implement an AAA client to interact with the AAA infrastructure, this
endpoint will have more resources. We describe two different
sequence flow. First, it is shown in Figure 2 where the AUTH option
is used at the end of EAP authentication. Second diagram (see
Figure 3) shows the flow when DTLS is used to protect CoAP messages
at the end of the EAP authentication. As an example, both diagrams
show the usage of the EAP-PSK method [RFC4764] as authentication
mechanism. (NOTE: any EAP method which is able to export
cryptographic material should be valid).

3.1. EAP in CoAP with AUTH option

If the EAP peer discovers the presence of the EAP authenticator and
wants to start the authentication, it can send a Non-Confirmable
"POST /auth" request to the node (Step 0). This message, will carry
an option developed from this work [RFC7967] called no response. The
rationale of these options is avoiding having to listen to a response
if is not needed. So the use of this option will allow us to signal
the intention of the EAP-Peer to start the authentication process.
Immediately after that, the EAP authenticator will start the
authentication service. It is worth noting that the EAP
authenticator may decide to start the authentication without waiting
for a "POST /auth" message.

In any case, to perform the authentication service, the CoAP client
(EAP authenticator) sends a Confirmable "POST /auth" request to the
CoAP Server (Step 1). POST message indicates to the CoAP server the
creation of a resource for the EAP-based authentication service. The
CoAP server assigns a resource and answers with an Acknowledgment
with the piggy-backed resource identifier (Uri-Path) (Step 2). It is
assumed that the CoAP server will only have an ongoing authentication
and will not process simultaneous EAP authentications in parallel to
save resources. Moreover if after a period of time (TBD) no further
message is received from the CoAP client, the CoAP server will free
the created state. In this moment, the CoAP server has started a
resource for the EAP authentication, whose resource identifier value
will be used together with the Token option value to relate all the
EAP conversation between both CoAP endpoints.

From now on, the EAP authenticator and the EAP peer will exchange EAP
packets transported in the CoAP message payload (Steps
3,4,5,6,7,8,9). The EAP authenticator will use POST method to send
EAP requests to the EAP peer. The EAP peer will use a Piggy-backed
response in the Acknowledgement message to carry the EAP response.
At the end of the message exchanges, if everything has gone well, the
EAP authenticator is able to send an EAP Success message and both
CoAP endpoints will share a Master Session Key (MSK) ([RFC5295])

   If the new defined AUTH option is used, an authentication tag is
   generated with a new key named COAP_PSK, derived from the MSK. The
   Acknowledgment message from the CoAP server will also include an AUTH
   option so that the CoAP client can verify that the CoAP server
   obtained the MSK. This is shown in Steps 9 and 10. From that point
   any exchange (for example, Steps 11 and 12) between both CoAP
   endpoints are protected with the AUTH option. Finally, the CoAP
   client MAY send a Confirmable DELETE request to remove all the state
   related with the authentication service in the CoAP server (Steps 13
   and 14). The CoAP server may decide to remove the state after period
   of time in case not receiving a DELETE request. This may be easier
   if the EAP authenticator sends a session lifetime option (TBD) in the
   Step 9 (where the EAP Success is sent).

   On the contrary, if DTLS is used (see Figure 3), a DTLS_PSK is
   derived from the MSK. Moreover, exchanges between both CoAP
   endpoints are protected with DTLS from that point.

```
              EAP peer                          EAP Auth.
            (COAP server)                      (COAP client)
           -------------                      -------------
               |                                   |
               | NON [0x6af5]                      |
               | No-Response (0x1A)                |
         0)    | (Token 0xFA5B45FF4723BB43)        |
               | POST /auth                        |
               |---------------------------------->|
               |                                   |
               |                                   |
               |                       CON [0x73DE] |
               |             (Token 0x78728FD4AC3190FF) |
               |                           POST /auth |
               |                       Payload nonce_c |
         1)    |<----------------------------------|
               |                                   |
               | ACK [0x73DE]                      |
               | (Token 0x78728FD4AC3190FF)        |
               | 2.01 Created                      |
               | Uri-Path [auth/5]                 |
               | Payload nonce_s                   |
         2)    |---------------------------------->|
               |                                   |
               |                       CON [0x7654] |
               |             (Token 0x78728FD4AC3190FF) |
               |                   Payload EAP Req/Id |
               |                         POST /auth/5 |
```

```
              3) |<---------------------------------------|
                 |                                        |
                 | ACK [0x7654]                           |
                 | (Token 0x78728FD4AC3190FF)             |
                 | 2.04 Changed                           |
                 | Payload EAP Res/Id                     |
              4) |--------------------------------------->|
                 |                                        |
                 |                            CON [0x7654] |
                 |              (Token 0x78728FD4AC3190FF) |
                 |                     Payload EAP-PSK MSG 1 |
                 |                             POST /auth/5 |
              5) |<---------------------------------------|
                 |                                        |
                 | ACK [0x7654]                           |
                 | (Token 0x78728FD4AC3190FF)             |
                 | 2.04 Changed                           |
                 | Payload EAP-PSK MSG 2                  |
              6) |--------------------------------------->|
                 |                                        |
                 |                            CON [0x9869] |
                 |              (Token 0x78728FD4AC3190FF) |
                 |                     Payload EAP-PSK MSG 3 |
                 |                             POST /auth/5 |
              7) |<---------------------------------------|
                 |                                        |
                 | ACK [0x9869]                           |
                 | (Token 0x78728FD4AC3190FF)             |
                 | 2.04 Changed                           |
                 | Payload EAP-PSK MSG 4                  |
              8) |--------------------------------------->|
              MSK |                                       | MSK
               |  |                            CON [0x7811] |  |
           COAP_PSK|               (Token 0x78728FD4AC3190FF) |COAP_PSK
                 |                             AUTH option |
                 |                     Payload EAP Success | (*)
                 |                             POST /auth/5 |
              9) |<---------------------------------------|
                 |                                        |
             (*) | ACK [0x7811]                           |
                 | (Token 0x78728FD4AC3190FF)             |
                 | AUTH option                            |
                 | 2.04 Changed                           |
             10) |--------------------------------------->|

                 .............

                 |                                        |
```

```
                              |                   CON [0x7511] |
                              |     (Token 0x55566AF7464646BC) |  (*)
                              |                    AUTH option |
                              |                      GET /temp |
                 11) |<-------------------------------------|
                     |                                        |
                     | ACK [0x7511]                           |
              (*)    | (Token 0x55566AF7464646BC)             |
                     | AUTH option                            |
                     | 2.05 Content                           |
                     | "22.5C"                                |
                 12) |------------------------------------->|
                     ...............

                     |                                        |
                     |                   CON [0x7600] |
                     |     (Token 0x678443AA678BC690) | (*)
                     |                    AUTH option |
                     |                   DELETE /auth/5 |
                 13) |<-------------------------------------|
                     |                                        |
                     | ACK [0x7500]                           |
              (*)    | (Token 0x678443AA678BC690)             |
                     | AUTH option                            |
                     | 2.02 Deleted                           |
                 14) |------------------------------------->|


                     (*) Protected with AUTH option
```
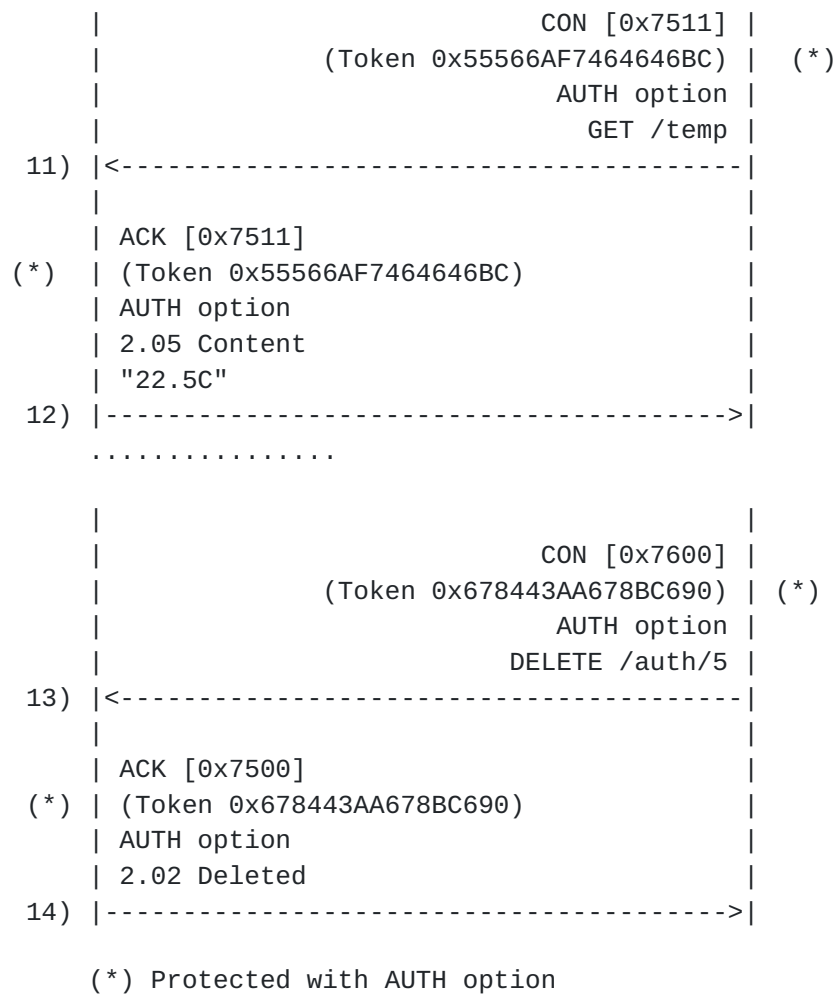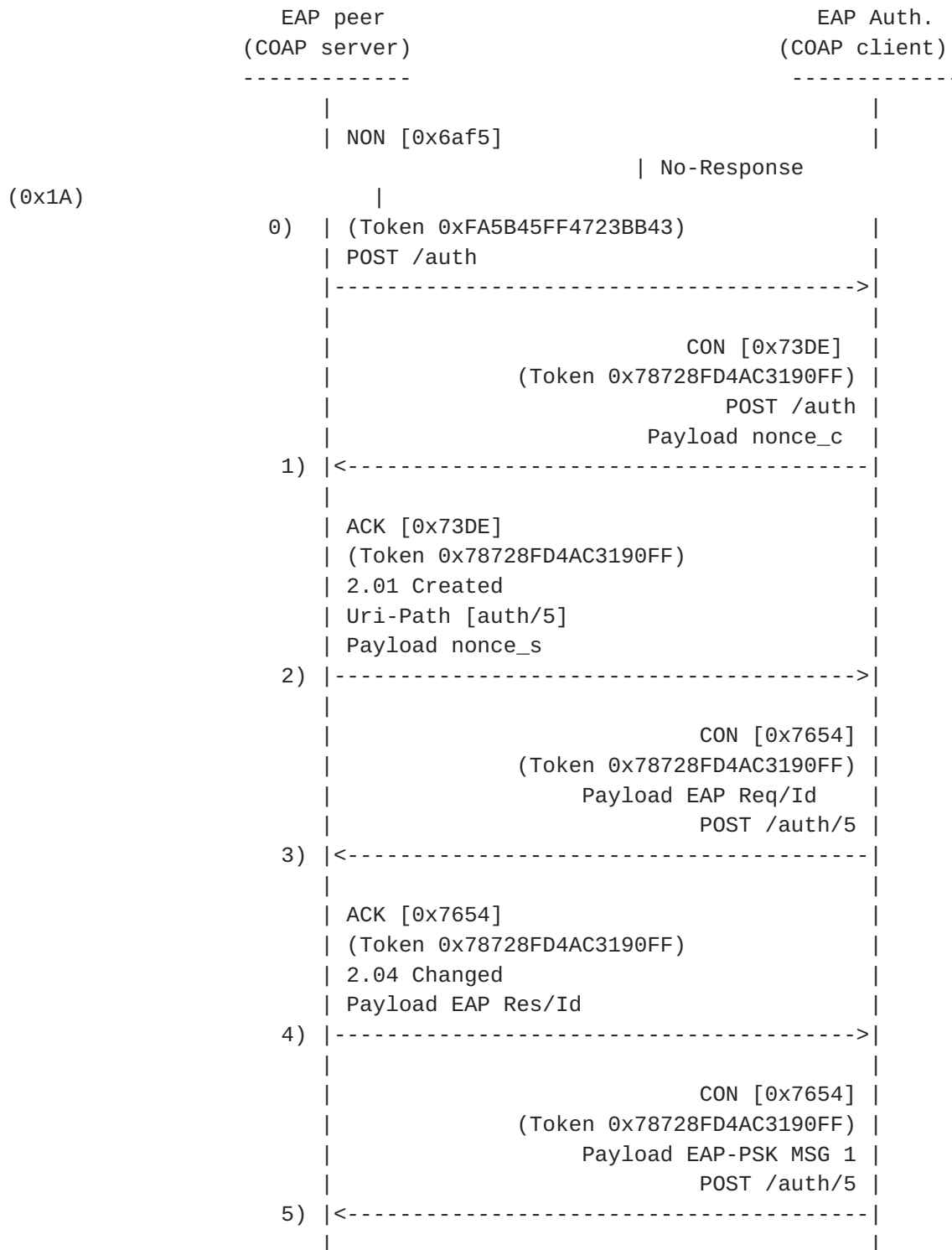
Figure 2: CoAP-EAP with AUTH option

3.2. CoAP-EAP with DTLS

   Other possibility at our disposal is to do a DTLS handshake after the
   MSKs generation and continue the communication between endpoints
   using CoAP through DTLS as we can see at Figure 3. The Steps 0-8 are
   the same as the case with AUTH option, however, before continuing
   with Steps 10 and 11, the EAP authenticator starts the DTLS handshake
   with the EAP peer (Step 9'). To establish a DTLS Security
   Association, a key named DTLS-PSK is derived from MSK (see Section 4
). In this case the CoAP client can start DTLS before sending the
   last message containing the EAP Success. Once DTLS is established,
   any posterior CoAP exchange is protected. Thus, new AUTH option is
   not needed. A successful DTLS negotiation confirms the possession of

DTLS_PSK that, in turn, corroborates that both entities participated
in the EAP authentication.

```
                    EAP peer                        EAP Auth.
                  (COAP server)                   (COAP client)
                  -------------                   -------------
                       |                               |
                       | NON [0x6af5]                  |
                                          | No-Response
(0x1A)                     |
                 0)  | (Token 0xFA5B45FF4723BB43)     |
                       | POST /auth                    |
                       |------------------------------->|
                       |                               |
                       |                  CON [0x73DE] |
                       |          (Token 0x78728FD4AC3190FF) |
                       |                    POST /auth |
                       |                 Payload nonce_c |
                 1)  |<-------------------------------|
                       |                               |
                       | ACK [0x73DE]                  |
                       | (Token 0x78728FD4AC3190FF)    |
                       | 2.01 Created                  |
                       | Uri-Path [auth/5]             |
                       | Payload nonce_s               |
                 2)  |------------------------------->|
                       |                               |
                       |                  CON [0x7654] |
                       |          (Token 0x78728FD4AC3190FF) |
                       |             Payload EAP Req/Id |
                       |                  POST /auth/5 |
                 3)  |<-------------------------------|
                       |                               |
                       | ACK [0x7654]                  |
                       | (Token 0x78728FD4AC3190FF)    |
                       | 2.04 Changed                  |
                       | Payload EAP Res/Id            |
                 4)  |------------------------------->|
                       |                               |
                       |                  CON [0x7654] |
                       |          (Token 0x78728FD4AC3190FF) |
                       |             Payload EAP-PSK MSG 1 |
                       |                  POST /auth/5 |
                 5)  |<-------------------------------|
                       |                               |
```

```
                         | ACK [0x7654]                        |
```

```
           | (Token 0x78728FD4AC3190FF)            |
           | 2.04 Changed                          |
           | Payload EAP-PSK MSG 2                  |
        6) |-------------------------------------->|
           |                                       |
           |                           CON [0x9869] |
           |             (Token 0x78728FD4AC3190FF) |
           |                    Payload EAP-PSK MSG 3 |
           |                            POST /auth/5 |
        7) |<--------------------------------------|
           |                                       |
           | ACK [0x9869]                          |
           | (Token 0x78728FD4AC3190FF)            |
           | 2.04 Changed                          |
           | Payload EAP-PSK MSG 4                  |
        8) |-------------------------------------->|
       MSK |                                       | MSK
        |  |                                       | |
  DTLS_PSK |                                       | DTLS_PSK
           |                                       |
           |             DTLS HANDSHAKE            |
           |          (Initiated by EAP Auth.)     |
       9') |<-------------------------------------->|
           |                                       |
           |                           CON [0x7811] |
           |             (Token 0x78728FD4AC3190FF) |
           |                    Payload EAP Success | (*)
           |                            POST /auth/5 |
       10) |<--------------------------------------|
           |                                       |
           | ACK [0x7811]                          |
      (*)  | (Token 0x78728FD4AC3190FF)            |
           | 2.04 Changed                          |
       11) |-------------------------------------->|


           .............


           |                                       |
           |                           CON [0x7511] |
           |             (Token 0xAA763D82F623B7FF) | (*)
           |                              GET /temp |
       12) |<--------------------------------------|
           |                                       |
           | ACK [0x7511]                          |
      (*)  | (Token 0xAA763D82F623B7FF)            |
           | 2.05 Content                          |
           | "22.5C"                               |
       13) |-------------------------------------->|
```

```
                           ...............

                  |                                          |
                  |                            CON [0x7600] |
                  |              (Token 0x678443AA678BC690) | (*)
                  |                            DELETE /auth/5 |
             14)  |<-----------------------------------------|
                  |                                          |
                  | ACK [0x7500]                             |
            (*)  | (Token 0x678443AA678BC690)                |
                  | 2.02 Deleted                             |
             15)  |----------------------------------------->|


                  (*) Protected with DTLS
```


Figure 3: EAP over CoAP with DTLS

3.3. CoAP as EAP lower-layer

   In this section we discuss the suitability of the CoAP protocol as
   EAP lower layer, and review the requisites imposed by the EAP
   protocol to any protocol that transports EAP. The assumptions EAP
   makes about its lower layers can be found in section 3.1 of the rfc
   [RFC3748], which are enumerated next:

   o Unreliable transport. EAP does lower layers are not assumed
      reliable.

   o Lower layer error detection. EAP relies on lower layer error
      detection (e.g., CRC, Checksum, MIC, etc.)

   o Lower layer security. EAP does not require security services from
      the lower layers.

   o Minimum MTU. Lower layers need to provide an EAP MTU size of 1020
      octets or greater.

   o Possible duplication. EAP stipulates that, while desirable, it
      does not require for the lower layers to provide non-duplication.

   o Ordering guarantees. EAP relies on lower layer ordering
      guarantees for correct operation.

   Next we go over the previous points to verify that CoAP does fits the
   EAP lower layer requirements. Regarding the unreliable transport,
   although EAP assumes a non reliable transport, CoAP does provide a

reliability mechanism through the use of Confirmable messages. For
the error detection, CoAP goes on top of UDP which provides a
checksum mechanism over its payload. Lower layer security services
are not required. About the minimum MTU of 1020 octets, CoAP assumes
an upper bound of 1024 for its payload which covers the requirements
of EAP. About possible duplication, although not required, CoAP
provides a message-ID value for deduplication purposes. Finally for
the ordering guarantees needed by EAP, CoAP message-ID can be used
for this purpose.

As we can see, CoAP does fulfill the requirements of EAP to be
considered suitable as lower-layer.

3.4. Optimization Considerations

Here we consider two possible optimizations for reducing the message
length:

o A first optimization would be to reduce the URI of the
   bootstrapping service. For example, the /auth URI could be
   reduced to /a.

o Another optimization would be to use a zero length CoAP token in
   the exchange.

Some use cases as LoRA Networks [I-D.pelov-core-cosol] might take
advantage of these reductions to improve the performance of the
bootstrapping process.

4. Key Derivation for protecting CoAP messages

As a result of a successful EAP authentication, both CoAP server and
CoAP client share a Master Key Session (MSK). The assumption is that
MSK is a fresh key so any derived key from the MSK will be also
fresh. We have considered the derivation of either COAP_PSK or
DTLS_PSK.

4.1. Deriving COAP_PSK

A key COAP_PSK is derived from the MSK to generate the authentication
tag included in the AUTH option. COAP_PSK is derived by using AES-
CMAC-PRF-128 [RFC4615], which, in turn, uses AES-CMAC-128 [RFC4493].
In this case, rest of CoAP exchanges between both entities can be
protected with integrity (NOTE: encryption will be considered in the
future) with AUTH option without the need of using DTLS. Thus, all
CoAP messages MUST include AUTH option from that point. (NOTE: We
understand that this would not be the standard way of protecting CoAP
but instead a new way of protecting CoAP messages).

COAP_PSK is a 16-byte length key which is computed in the following way:

COAP_PSK = KDF(MSK, "IETF_COAP_PSK" || nonce_c || nonce_s, 64, length)

where:

o The AES-CMAC-PRF-128 is defined in [RFC4615]. This function uses AES-CMAC-128 as building block.

o The MSK exported by the EAP method.

o "IETF_COAP_PSK" is the ASCII code representation of the non-NULL terminated string (excluding the double quotes around it). This value is concatenated with the value of the Token Option value.

o nonce_c is the random value sent by the EAP Authenticator to the EAP Peer in the POST Message.

o nonce_s is the random value sent by the EAP Peer to the EAP Authenticator in the Acknowledgment to the POST Message.

o 64 is the length of the MSK.

o length is the length of the label "IETF_COAP_PSK" (13 bytes).

4.2. Deriving DTLS_PSK

In the second alternative, a DTLS_PSK is derived from the MSK between both CoAP endpoints. So far, DTLS_PSK will have also 16 byte length and it will derived as follows:

DTLS_PSK = KDF(MSK, "IETF_DTLS_PSK" || nonce_c || nonce_s, 64, length). This value is concatenated with the value of the Token Option value.

where:

o MSK is exported by the EAP method.

o "IETF_DTLS_PSK" is the ASCII code representation of the non-NULL terminated string (excluding the double quotes around it).

o nonce_c is the random value sent by the EAP Authenticator to the EAP Peer in the POST Message.

   o nonce_s is the random value sent by the EAP Peer to the EAP
      Authenticator in the Acknowledgment to the POST Message.

   o 64 is the length of the MSK.

   o length is the length of the label "IETF_DTLS_PSK" (13 bytes).

   As mentioned in [RFC4279], a PSK identity is needed. We are
   considering the usage of the Token Option value chosen during the EAP
   authentication as identity. In any case, this still needs further
   investigation.

5. Generating AUTH option

   A new AUTH option is defined in this document for authentication
   purposes. Following guidelines in [RFC7252] this option is:

   1. Format opaque (sequence of bytes).

   2. Elective.

   3. Unsafe to Forward.

   4. No cacheable.

   The number of the option will be determined by this previous
   decisions.

   1. Elective (C = 0)

   2. Unsafe to Forward (0)

   3. NoCacheKey (111)

   The number of the AUTH option will fit this pattern: xxx11100

```
                    0   1   2   3   4   5   6   7
                   +---+---+---+---+---+---+---+---+
                   |           | NoCacheKey| U | C |
                   +---+---+---+---+---+---+---+---+
```

                   Figure 4: Auth Option Number Mask

   First available option number is 01011100 (92).

   The resultant AUTH option is:

```
                +-----+----+---+---+---+----------+--------+--------
+---------+
                | No. | C  | U | N | R | Name     | Format | Length |
Default |
                +-----+----+---+---+---+----------+--------+--------
+---------+
                | 92  |    |   | x |   | AUTH     | opaque | 128    |
(none)   |
                +-----+----+---+---+---+----------+--------+--------
+---------+
```

Figure 5: AUTH Option

C, U, N and R columns indicate the properties, Critical, UnSafe,
NoCacheKey and Repeatable, respectively.

To generate the value of the AUTH option, we use AES-CMAC-128 as
authentication algorithm. Thus, the AUTH option content will have an
authentication tag of 16 bytes.

AUTH Option value = AES-CMAC-128(COAP_PSK, MSG, MSG_LENGTH)

where:

o COAP_PSK is the key derived in the CoAP Security Association
   process.

o MSG is the CoAP message including AUTH option filled with zeros.

o MSG_LENGTH. Length of the CoAP message.

After applying AES-CMAC-128 function, the AUTH option value will be
set in the AUTH option replacing the zeros.

6. Implementation

At the time of writing this document, we have developed a proof-of-
concept based on libcoap ([libCoAP]) in PC platform and started the
development of a simulation with COOJA network simulator for Contiki
([Contiki]).

So far, we have implemented an authentication tag by using AES-CMAC-
128. However this authentication tag has been included in the
payload of two final messages after sending the EAP Success. The
implementation of the AUTH option will come soon. Moreover, we have
used AES-CMAC-128 for COAP_PSK. Since this function does not allow a
key longer than 16 bytes, we have used the most significative 16
bytes of the MSK as input key. Since AES-CMAC-PRF-128 eliminates
this limitation, we will implement this version instead.

We are using (for the PC version) libeap in wpa-supplicant and
hostapd open source software ([hostapd]) to implement the EAP stack
and, in particular, the EAP-PSK method.

7. Future Work: CoAP Intermediaries

Architecture explained in Figure 1 assumes that EAP peer can
communicate with the EAP authenticator. In certain scenarios, EAP
authenticator may not be reachable from the EAP peer if the EAP
authenticator is placed several hops-away. In this scenario,
described in Figure 6, we are considering the usage a new service
that assists the authentication. This service acts as a intermediary
of CoAP messages between the EAP peer and EAP authenticator.
Currently we have a design of three different variants of this
entity.

o CoAP relay

o CoAP proxy

o CoAP stateless proxy

Proof-of-concept implementations of the CoAP relay and CoAP proxy and
CoAP stateless proxy are evaluated. The strategy is similar to the
one described in PANA Relay ([RFC6345]) or DTLS Relay
([I-D.kumar-dice-dtls-relay]). Unlike CoAP proxy, the CoAP relay is
not intended to keep any state (stateless behavior) and the EAP peer
is not assumed to be aware of the presence of the CoAP relay. The
CoAP stateless proxy provides an approach between the previous two.
It behaves as a proxy, but avoid generating any state related to the
ongoing exchange.

```
               +------------+        +------------+        +--------------+
               | EAP peer/  |        |    CoAP    |        |  EAP auth    |
               | CoAP server|+------+|intermediary|+------+|  CoaP client |
               +------------+  CoAP  +------------+  CoAP  +--------------+
               |
          AAA  |
               |
               +--------------+
               |  EAP server/ |
               |  AAA server  |
               +--------------+
```
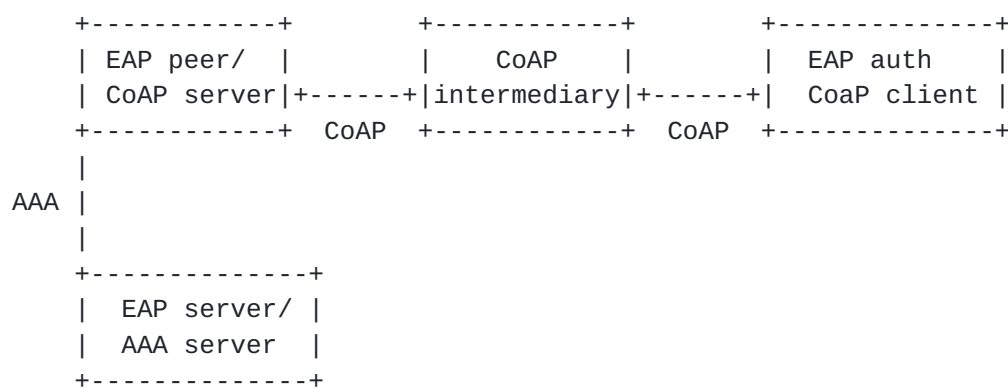
Figure 6: CoAP EAP Intermediary Architecture

Once the EAP peer has been authenticated, CoAP intermediary should
not be needed anymore for this EAP peer.

Development of this new service may modify the "Unsafe to Forward"
flag of the AUTH option.

8. Use Case Scenario

In the following, we explain a basic example about the usage of CoAP-
EAP. There are 5 entities involved in the scenario:

o 2 nodes (A and B), which are constrained devices.

o 1 node D, which is considered a no so constrained device, such as
   a phone, or a tablet or even a laptop.

o 1 controller (C). The controller manages a domain where nodes can
   be deployed. It can be considered a more powerful machine than
   the nodes, however it may have some constrained resources.

o 1 AAA server (AAA). The AAA is an Authentication, Authorization
   and Accounting Server, which is not constrained.

Any node wanting to join the domain managed by the controller, must
perform an CoAP-EAP authentication with the controller C. This
authentication, as depicted in Figure 6, may involve an external AAA
server. This means that A and B, once deployed, will perform this
CoAP-EAP once as a bootstrapping phase to establish a security
association with the controller C. Moreover, any other entity (i.e.
node D) , which wants to join and establish communications with nodes
under the controller C's domain must also do the same.

One use case is the following. The node A wants to communicate with
node B (e.g. to active a light switch). The overall process is
divided in three phases. Let's start with node A. In the first
phase, the node A (EAP peer) does not yet belong to the controller
C's domain. Then, it communicates with controller C (EAP
authenticator) and authenticates with CoAP-EAP, which, in turn,
communicates with the AAA server to complete the authentication
process. If the authentication is successful, key material is
distributed to the controller C and derived by node A. This key
material allows node A to establish a security association with
controller C. Some authorization information may be also provided in
this step. If authentication and authorization are correct, node A
is enrolled in the controller C's domain during a period of time. In
particular, [RFC5247] recommends 8 hours, though the AAA server can
establish this lifetime. In the same manner, B needs to perform the
same process with CoAP-EAP to be part of the controller C's domain.

In the second phase, when node A wants to talk with node B, it
contacts the controller C for authorization to access node B and

obtain all the required information to do that in a secure manner
(e.g. keys, tokens, authorization information, etc.). It does not
require the usage of CoAP-EAP. The details of this phase are out of
scope of this document.

In the third phase, the node A can access node B with the credentials
and information obtained from the controller C in the second phase.
This access can be repeated without contacting the controller, while
the credentials given to A are still valid. The details of this
phase are out of scope of this document.

It is worth noting that first phase with CoAP-EAP is only required to
join the controller C's domain. Once it is performed with success,
the communications are local to the controller C's domain so there is
no need to contact the external AAA server.

Another use case is the following. Node D wants to communicate with
node A (e.g. to obtain a temperature measurement). To do that, first
of all, node D must join the controller C's domain. To do that it
performs a CoAP-EAP authentication and authorization with the
controller C (first phase). If everything ends with success, the
node D can request access to node A to C (second phase). Then if
node D is authorized can access to node A (third phase). So, in the
end, node D also implements CoAP-EAP as any other constrained node.

9. Acknowledgments

We would like to thank Pedro Moreno-Sanchez and Gabriel Lopez-Millan
for the first review of this document. Also, we would like to thank
Ivan Jimenez-Sanchez for the first proof-of-concept implementation of
this idea.

We also thank for their valuables comments to Alexander Pelov and
Laurent Toutain, specially for the potential optimizations of CoAP-
EAP.

This work has been partly funded by European Commission through the
FP7-SMARTIE-609062 EU Project.

10. Security Considerations

There are some aspects to be considered such as how authorization is
managed, how the cryptographic suite is selected and how the trust in
the Controller is established.

10.1. Authorization

   Authorization is part of the bootstrapping. It serves to establish
   whether the node can join and the set of conditions it has to adhere.
   The authorization data received from the AAA server (RADIUS in this
   case) can be delivered in RADIUS attributes such as NAS-Filter-Rules,
   Framed-MTU, Session-Timeout, etc. Providing more fine grained
   authorization data can be with the transport of SAML in RADIUS
   [RFC7833] After bootstrapping, additional authorization to operate in
   the security domain, e.g., access services offered by other nodes,
   can be taken care of by the solutions proposed in the ACE WG.

10.2. Cryptographic suite selection

   How the cryptographic suit is selected is also important. To reduce
   the overhead of the protocol we use a default cryptographic suite.
   To support the protocol all implementations MUST at least support
   AES-CMAC-PRF-128 as the KDF and AES-CMAC-128 as default. The
   cryptographic suite is not negotiated. If the cryptographic suite to
   be used by the node is different from default, the AAA server will
   send the specific parameters to the Authenticator. If the
   cryptographic suite is not supported, the key derivation process
   would result in a security association failure.

10.3. Additional Security Consiederation

   Other security related concerns can be how to ensure that the node
   joining the security domain can in fact trust the Controller. This
   issue is elaborated in the EAP KMF[RFC5247] . Summarizing, the node
   knows it can trust the Controller, because the key that is used to
   establish the security association is derived from the MSK. If the
   Controller has the MSK, it is clear the AAA Server of the node trusts
   the Controller, which confirms it is a trusted party.

11. IANA Considerations

   This document has no actions for IANA.

12. References

12.1. Normative References

   [I-D.kumar-dice-dtls-relay]
             Kumar, S., Keoh, S., and O. Garcia-Morchon, "DTLS Relay
             for Constrained Environments", draft-kumar-dice-dtls-
             relay-02 (work in progress), October 2014.

   [I-D.ohba-core-eap-based-bootstrapping]
              Das, S. and Y. Ohba, "Provisioning Credentials for CoAP
              Applications using EAP", draft-ohba-core-eap-based-
              bootstrapping-01 (work in progress), March 2012.

   [I-D.pelov-core-cosol]
              Pelov, A., Toutain, L., and Y. Delibie, "Constrained
              Signaling Over LP-WAN", draft-pelov-core-cosol-01 (work in
              progress), February 2016.

   [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
              Requirement Levels", BCP 14, RFC 2119,
              DOI 10.17487/RFC2119, March 1997,
              <https://www.rfc-editor.org/info/rfc2119>.

   [RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H.
              Levkowetz, Ed., "Extensible Authentication Protocol
              (EAP)", RFC 3748, DOI 10.17487/RFC3748, June 2004,
              <https://www.rfc-editor.org/info/rfc3748>.

   [RFC4279] Eronen, P., Ed. and H. Tschofenig, Ed., "Pre-Shared Key
              Ciphersuites for Transport Layer Security (TLS)",
              RFC 4279, DOI 10.17487/RFC4279, December 2005,
              <https://www.rfc-editor.org/info/rfc4279>.

   [RFC4493] Song, JH., Poovendran, R., Lee, J., and T. Iwata, "The
              AES-CMAC Algorithm", RFC 4493, DOI 10.17487/RFC4493, June
              2006, <https://www.rfc-editor.org/info/rfc4493>.

   [RFC4615] Song, J., Poovendran, R., Lee, J., and T. Iwata, "The
              Advanced Encryption Standard-Cipher-based Message
              Authentication Code-Pseudo-Random Function-128 (AES-CMAC-
              PRF-128) Algorithm for the Internet Key Exchange Protocol
              (IKE)", RFC 4615, DOI 10.17487/RFC4615, August 2006,
              <https://www.rfc-editor.org/info/rfc4615>.

   [RFC4764] Bersani, F. and H. Tschofenig, "The EAP-PSK Protocol: A
              Pre-Shared Key Extensible Authentication Protocol (EAP)
              Method", RFC 4764, DOI 10.17487/RFC4764, January 2007,
              <https://www.rfc-editor.org/info/rfc4764>.

   [RFC5247] Aboba, B., Simon, D., and P. Eronen, "Extensible
              Authentication Protocol (EAP) Key Management Framework",
              RFC 5247, DOI 10.17487/RFC5247, August 2008,
              <https://www.rfc-editor.org/info/rfc5247>.

   [RFC5295] Salowey, J., Dondeti, L., Narayanan, V., and M. Nakhjiri,
              "Specification for the Derivation of Root Keys from an
              Extended Master Session Key (EMSK)", RFC 5295,
              DOI 10.17487/RFC5295, August 2008,
              <https://www.rfc-editor.org/info/rfc5295>.

   [RFC6345] Duffy, P., Chakrabarti, S., Cragie, R., Ohba, Y., Ed., and
              A. Yegin, "Protocol for Carrying Authentication for
              Network Access (PANA) Relay Element", RFC 6345,
              DOI 10.17487/RFC6345, August 2011,
              <https://www.rfc-editor.org/info/rfc6345>.

   [RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained
              Application Protocol (CoAP)", RFC 7252,
              DOI 10.17487/RFC7252, June 2014,
              <https://www.rfc-editor.org/info/rfc7252>.

   [RFC7833] Howlett, J., Hartman, S., and A. Perez-Mendez, Ed., "A
              RADIUS Attribute, Binding, Profiles, Name Identifier
              Format, and Confirmation Methods for the Security
              Assertion Markup Language (SAML)", RFC 7833,
              DOI 10.17487/RFC7833, May 2016,
              <https://www.rfc-editor.org/info/rfc7833>.

   [RFC7967] Bhattacharyya, A., Bandyopadhyay, S., Pal, A., and T.
              Bose, "Constrained Application Protocol (CoAP) Option for
              No Server Response", RFC 7967, DOI 10.17487/RFC7967,
              August 2016, <https://www.rfc-editor.org/info/rfc7967>.

12.2. Informative References

   [Contiki] "Contiki: The Open Source OS for the Internet of Things",
              March 2014.

   [hostapd] "hostapd: IEEE 802.11 AP, IEEE 802.1X/WPA/WPA2/EAP/RADIUS
              Authenticator", March 2014.

   [libCoAP] "C-Implementation of CoAP", January 2013.

Authors' Addresses

   Rafa Marin-Lopez
   University of Murcia
   Campus de Espinardo S/N, Faculty of Computer Science
   Murcia 30100
   Spain

   Phone: +34 868 88 85 01
   Email: rafa@um.es

   Dan Garcia Carrillo
   University of Murcia
   Campus de Espinardo S/N, Faculty of Computer Science
   Murcia 30100
   Spain

   Phone: +34 868 88 78 82
   Email: dan.garcia@um.es